

ЗАПИТИ З МНОЖИННИМИ ПОРІВНЯННЯМИ У МОБІ SQL (СУБД MYSQL)

У статті досліджуються схеми реалізації складних запитів з множинними порівняннями засобами мови SQL без вкладених підзапитів, що характерно для деяких систем баз даних, зокрема СУБД MySQL.

Мова SQL на сьогоднішній день є найпоширенішою мовою запитів СУБД реляційного типу і підтримується практично всіма програмними продуктами, які представлені на ринку технологій, пов'язаних з базами даних та інформаційними системами.

У [6] наводиться кілька реалізаційних схем для запитів з множинними порівняннями, але істотною особливістю цих прикладів є орієнтація на можливість використання багаторівневих SELECT-операторів (або SELECT-операторів з фразами SubSELECT). Водночас деякі СУБД, зокрема MySQL [5], підтримують версію мови SQL, де відсутні SubSELECT, тому викликають певну зацікавленість технологічні прийоми, спрямовані на реалізацію складних запитів з множинними порівняннями на основі однорівневих операторів SELECT.

Для розгляду прикладів типів запитів опишемо структуру таблиць з предметної області «Розклад вищого навчального закладу» (табл. 1–4). У наведених таблицях опущені описи доменів атрибутів, бо для подальших прикладів це несуттєво. Атрибути, що утворюють ключі відповідних реляцій, виділені підкресленням. Для зручності та кращого розуміння в таблицях наводяться як українські назви атрибутів і реляцій, так і більш звичні для мови SQL відповідні англійські назви.

Спочатку розглянемо запит:

«Знайти прізвища лекторів, які не проводять ніяких занять (тобто відсутні у розкладі)». Такий запит можна реалізувати операцією віднімання кодів лекторів з реляції Розклад від множини кодів лекторів з реляції Лектор.

Основним засобом, за допомогою якого таке завдання можна реалізувати в SQL-92, є операція EXCEPT.

```
(SELECT Lect.CdL
FROM Lect) EXCEPT
(SELECT Sch.CdL
FROM Sch);
```

У версії SQL (MS Access) операція EXCEPT відсутня, тому зазвичай використовуються такі варіанти:

```
(SELECT Lect.CdL
FROM Lect
WHERE Lect.CdL NOT IN
(SELECT Sch.CdL
FROM Sch);
```

або

```
SELECT Lect.CdL
FROM Lect LEFT JOIN Sch ON Lect.CdL = Sch.CdL
WHERE Lect.CdL = NULL;
```

Як бачимо, в останньому випадку застосовується однорівневий SELECT, тому така схема

Таблиця 1

Лектор	Код лектора	Прізвище	Організація	Телефон	Науковий ступінь
Lect	CdL	Name	Org	Phone	Degree

Таблиця 2

Предмет	Код предмета	Назва	Кількість годин	Тип	Контроль
Subj	CdS	Nm	Hours	Tr	Testing

Таблиця 3

Група	Код групи	Курс	Кафедра	Факультет	Кількість чол.
Grp	CdG	Course	Caf	Faculty	Quant

Таблиця 4

Розклад	Код лектора	Код предмета	Код групи	Аудиторія	День	Пара	Тиждень
Sch	CdL	CdS	CdG	Aud	Day	TmL	Week

реалізації допустима і в СУБД, де підтримуються тільки однорівневі оператори, наприклад MySQL.

Підхід до вирішення проблеми множинного порівняння може базуватися на тій обставині, що з усіх операцій реляційної алгебри Кодда тільки операція ділення не дає змогу безпосереднього представлення засобами SQL-92 (чи SQL Access). Але операція ділення не є незалежною, тобто вона може бути виражена через інші операції реляційної алгебри. Зокрема, в [4] наводиться відповідна формула.

Якщо задані дві реляції $A[X,Y]$ і $B[Y]$, то

$$(A[Y \div Y]B) \equiv (A[X] \setminus ((A[X] \otimes B) \setminus A[X])),$$

де \div – операція ділення; $[]$ – операція проєкції; \setminus – операція різниці; \otimes – декартів добуток.

Усі операції, що входять до правої частини тотожності, допускають безпосереднє представлення засобами SQL-92, а також однорівневи операторами SELECT.

```

Q1:      SELECT A.X
{A[X]}   FROM A;
Q2:      SELECT Q1.X, B.Y
{A[X]  $\otimes$  B} FROM Q1,B;
Q3:      SELECT Q2.X
{(A[X]  $\otimes$  B)  $\setminus$  A[X]} FROM Q2 LEFT JOIN A ON
                                Q2.X=A.X AND Q2.Y=A.Y
                                WHERE A.X = NULL AND A.Y
                                = NULL;
Q4:      SELECT A.X
{A[X] \ Q3} FROM A LEFT JOIN Q3 ON
                                A.X = Q3.X
                                WHERE Q3.X = NULL;
```

Розглянемо тип запиту, що пропонується в [4]: «Знайти прізвища лекторів, що читають усі предмети».

Для реалізації такого типу запиту засобами однорівневого SQL декомпонуємо його на ряд підзапитів:

1. Знайти всі можливі варіанти читання предметів викладачами (Q1)

```

SELECT Sch.CdL, S0ubj.CdS
FROM Sch, Subj;
```

2. Знайти коди лекторів, що не читають якийсь предмет (Q2)

```

SELECT Q1.CdL
FROM Q1 LEFT JOIN Sch ON (Q1.CdL = Sch.CdL)
                                AND (Q1.CdS = Sch.CdS)
WHERE (Sch.CdL = NULL) AND (Sch.CdS = NULL);
```

3. Знайти коди лекторів, які читають усі предмети (Q3)

```

SELECT Sch.CdL
FROM Q2 RIGHT JOIN Sch ON (Q2.CdL = Sch.CdL)
WHERE (Q2.CdL = NULL);
```

3. Знайти прізвища лекторів, які читають усі предмети (Q4)

```

SELECT Lect.Name
FROM Lect INNER JOIN Q3 ON (Q3.CdL = Lect.CdL);
```

Розглянемо ще один запит, дещо складніший: «Знайти прізвища лекторів, які читають принаймні всі ті предмети, які читає викладач Іванчук».

У термінах класичного SEQUEL цей запит може виглядати, наприклад, так:

```

SELECT DISTINCT Lect.Name
FROM Lect
WHERE Lect.CdL IN
  (SELECT Sch.CdL
   FROM Sch
   GROUP BY Sch.CdL
   HAVING SET(Sch.CdS)
           CONTAINS //  $\supseteq$ 
  (SELECT Sch.CdS
   FROM Sch
   WHERE Sch.CdL IN
     SELECT Lect.CdL
     FROM Lect
     WHERE Lect.Name = «Іванчук»));
```

Для представлення цього запиту за допомогою однорівневих операторів SELECT декомпонуємо початковий запит на підзапити.

1. Знайти коди предметів, які читає Іванчук (Q1)

```

SELECT Sch.CdS
FROM Sch INNER JOIN Lect ON
  Sch.CdL = Lect.CdL
WHERE Lect.Name = «Іванчук»;
```

```

(Q2)
SELECT Sch.CdL, Q1.CdS
FROM Sch, Q1;
```

2. Знайти коди лекторів, що не читають якийсь із тих предметів, які читає Іванчук (Q3)

```

SELECT Sch.CdL
FROM Q2 LEFT JOIN Sch ON
  Sch.CdL = Q2.CdL AND Sch.CdS = Q2.CdS
WHERE Sch.CdL = NULL AND Sch.CdS = NULL;
```

3. Знайти коди лекторів, які читають принаймні всі ті предмети, які читає Іванчук (Q4)

```

SELECT Sch.CdL
FROM Q3 RIGHT JOIN Sch ON
  Sch.CdL = Q3.CdL AND Sch.CdS = Q3.CdS
WHERE Q3.CdL = NULL AND Q3.CdS = NULL;
```

4. Знайти прізвища лекторів попереднього запиту (Q5)

```

SELECT Lect.Name
FROM Q4 INNER JOIN Lect ON
  Q4.CdL = Lect.CdL ;
```

Наприкінці зауважимо, що можливість підтримувати складні запити з множинними порівняннями тією чи іншою *версією* мови SQL має велике теоретичне значення, адже відсутність відповідних засобів означає втрату властивості *реляційної повноти* [4]. У практичній роботі складні запити з множинними порівняннями займають відносно невелику частку у загальній

сукупності всіх запитів, але виразити їх простими типами запитів (не виходячи за межі мови SQL) не вдається, і тоді користувачі отримують від розробників рекомендації про *автоматизовану* (а не автоматичну) селекцію потрібної інформації, що звичайно можливо лише при відносно невеликих об'ємах даних, що відбираються.

На основі проведеного дослідження можна зробити висновок, що версія SQL (MySQL) не є реляційно повною, але сама система MySQL за рахунок підтримки використання попередньо виконаних запитів та системного зберігання їх результатів задовольняє властивості реляційної повноти.

1. Chamberlin D. D., Boyce R. F. SEQUEL: A Structured English Query Language II Proc. ACM SIGMOD Workshop on Data Description, Access and Control- Ann Arbor, Mich., 1974.
2. Chamberlin D. D. et al. SEQUEL/2: A Unified Approach to Data Definition, Manipulation, and Control II IBM J. R&D- 1976.-20, № 6, - 1977.-21, № 1.
3. Date C J., Darven H. A Guide to the SQL Standard- Reading, Mass.: Addison-Wesley, 1993.
4. Деят К. Дж. Введение в системы баз данных, - 6-е изд.- К.: Диалектика, 1998.
5. www.mysql.com.
6. Глибовець М. М., Кулябко П. П. Запити з множинними порівняннями у мові SQL.- Наукові записки НаУКМА- Т. 18.- Комп'ютерні науки-2000-С 19-21.
7. Кулябко П. П., Пешко О. І., Римарчук В. К. Ефективність виконання запитів з множинними порівняннями у мові SQL.- Наукові записки НаУКМА. Т. 19-20- Комп'ютерні науки-2002-С 60-64.

5. S. Gorohovskiy, P. P. Kuliabko, O. P. Kuliabko, O. V. Franchuk

QUERIES WITH SET COMPARISONS IN SQL (DBMS MYSQL).

Realization schemes of the complicated queries with set comparisons by SQL means without SubSelect clauses are investigated in this paper. Such SQL version is presented in some DBMS, for example, MySQL.